Requested Patent: JP6028199

Title: SYNCHRONISING CONCURRENT PROCESSES IN A COMPUTER SYSTEM.

Abstracted Patent: EP0445954

Publication Date: 1991-09-11

Inventor(s): ASHLAD YETURU (US); CHANG DANIEL T (US)

Applicant(s): IBM (US)

Application Number: EP19910301586 19910227

Priority Number(s): US19900488647 19900305

IPC Classification: G06F9/46

Equivalents:

ABSTRACT:

A method is provided for synchronising concurrent processes in a computer system. Each process has a virtual time. When messages are received and acted upon by a process, and such messages have a receive time greater than or equal to the current virtual time of the process, state consistency can be assumed. If quard checking is enabled, the guards preferably must still be satisfied before the message will be acted upon. If the time of receipt for the message is less than the current virtual time, further state consistency checks are made. Such consistency checks can include checking of variable versions and guards for consistency. If the process state is consistent with the contents of the message, the message can be processed even though the current virtual time is greater than the receive time of the message. The same general approach can also be used within a process having concurrent actions, with shared variables being checked for state consistency using virtual time stamps and variable versions.

Requested Patent: JP6028199

Title: SYNCHRONISING CONCURRENT PROCESSES IN A COMPUTER SYSTEM.

Abstracted Patent: EP0445954

Publication Date: 1991-09-11

Inventor(s): ASHLAD YETURU (US); CHANG DANIEL T (US)

Applicant(s): IBM (US)

Application Number: EP19910301586 19910227

Priority Number(s): US19900488647 19900305

IPC Classification: G06F9/46

Equivalents:

ABSTRACT:

A method is provided for synchronising concurrent processes in a computer system. Each process has a virtual time. When messages are received and acted upon by a process, and such messages have a receive time greater than or equal to the current virtual time of the process, state consistency can be assumed. If quard checking is enabled, the guards preferably must still be satisfied before the message will be acted upon. If the time of receipt for the message is less than the current virtual time, further state consistency checks are made. Such consistency checks can include checking of variable versions and guards for consistency. If the process state is consistent with the contents of the message, the message can be processed even though the current virtual time is greater than the receive time of the message. The same general approach can also be used within a process having concurrent actions, with shared variables being checked for state consistency using virtual time stamps and variable versions.

(19)日本国特許庁（ＪＰ）　(12) 公開特許公報（A）　(11)特許出願公開番号

特開平6－28199

(43)公開日　平成６年(1994)２月４日

(51)Int.Cl.[5]	識別記号	庁内整理番号	Ｆ１	技術表示箇所
Ｇ０６Ｆ　9/46	３４０　Ａ	8120－5Ｂ		

審査請求　有　　請求項の数22(全 12 頁)

(21)出願番号　特願平3－35390

(22)出願日　平成３年(1991)２月５日

(31)優先権主張番号　４８８６４７

(32)優先日　1990年３月５日

(33)優先権主張国　米国（ＵＳ）

(71)出願人　390009531
インターナショナル・ビジネス・マシーンズ・コーポレイション
ＩＮＴＥＲＮＡＴＩＯＮＡＬ　ＢＵＳＩＮＥＳＳ　ＭＡＳＣＨＩＮＥＳ　ＣＯＲＰＯＲＡＴＩＯＮ
アメリカ合衆国10504、ニューヨーク州　アーモンク　（番地なし）

(72)発明者　イエツル、アシュラド
アメリカ合衆国テキサス州、オースチン、ウエスト、シックスス、1632－デー

(74)代理人　弁理士　頓宮　孝一　（外５名）

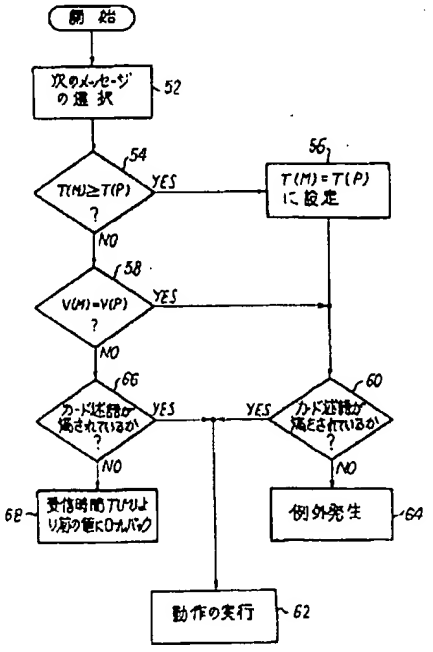
最終頁に続く

(54)【発明の名称】　並行処理の同期方法

(57)【要約】

【目的】　コンピュータシステムにおける並行処理を同期化するための方法を提供する。

【構成】　メッセージが受信され、処理によって作動される際に、そのようなメッセージが処理の現在の仮想時間よりも大きいか待ち行列それに等しい受信時間を有している場合、状態の一致性が仮定できる。ガード検査が使用可能となっていれば、ガードは好ましくは、そのメッセージが作動する以前に満たされていなければならない。メッセージの受信時間が現在の仮想時間よりも小さい場合、さらに状態の一致性検査が行われる。このような一致性検査は変数バージョンおよびガードの一致性の検査を含む。処理の状態がメッセージの内容と一致すれば、そのメッセージは、たとえ現在の仮想時間がメッセージの受信時間より大きくても、処理されることができる。同様の一般的方法は、仮想タイムスタンプおよび変数バージョンによって共用変数が状態の一致性を検査される、並行動作を有する１処理内でも使用することができる。



1

【特許請求の範囲】

【請求項1】 2処理間でメッセージを通信するための方法であって、

第1の処理から第2の処理へメッセージを転送するステップと、

そのメッセージのメッセージ時間を第2の処理の局所時間と比較するステップと、

そのメッセージ時間が第2の処理の局所時間よりも小さい場合はメッセージの内容が第2の処理の現在の状態と不一致であるどうかを判断するステップと、

メッセージの内容が不一致である場合は第2の処理をメッセージの内容と一致する状態までロールバックさせるステップと、

メッセージの内容が一致している場合はそのメッセージを処理するステップとを含む方法。

【請求項2】 請求項1記載の方法であって、判断するステップが、

第2の処理の処理変数バージョンを、メッセージを処理するために要求される変数バージョンと比較するステップと、

そのメッセージを処理するために要求される変数の変数バージョンが第2の処理の同一変数の現在のバージョンと異なる場合、そのメッセージの内容を第2の処理の現在の状態と不一致であると定義するステップとを含む方法。

【請求項3】 請求項2記載の方法であって、判断するステップはさらに、

そのメッセージを処理するために要求される変数バージョンが第2の処理の現在の処理変数バージョンと異なる場合、メッセージを処理するために要求される処理動作のためのガード述語の値を判定するステップと、

ガード述語が満たされない場合、そのメッセージの内容を第2の処理の現在の状態と不一致であると定義するステップとを含む方法。

【請求項4】 請求項1記載の方法であって、さらに、

そのメッセージ時間が第2の処理の処理時間よりも大きい場合、または、メッセージの内容が第2の処理の現在の状態と一致する場合、そのメッセージを処理するために必要な動作に関するガード述語の値を検査するステップを含む方法。

【請求項5】 請求項4記載の方法であって、さらに、

ガード述語の全部が必ずしも満たされない場合、当該ガード述語が満たされるまで不満足なガードを有するあらゆる動作の実行を遅延させるステップを含む方法。

【請求項6】 請求項1記載の方法であって、さらに、

そのメッセージ時間が第2の処理の処理時間よりも大きい場合、メッセージが処理される際に第2の処理の処理時間をメッセージの値に等しく設定するステップを含む方法。

【請求項7】 請求項1記載の方法であって、メッセージ

2

を処理するステップが、

第2の処理の1以上の処理ステップを実行するステップと、

結果のメッセージを第1の処理に戻すステップとを含む方法。

【請求項8】 請求項7記載の方法であって、結果メッセージが本来のメッセージのメッセージ時間の値に等しいメッセージ時間を有している方法。

【請求項9】 請求項1記載の方法であって、メッセージ時間が、第2の処理が当該メッセージを第2の処理のいずれかの仮想時間に処理できることを指示する既定の値を有する方法。

【請求項10】 請求項1記載の方法であって、第1の処理および第2の処理が目的向きシステムにおける目的を含む方法。

【請求項11】 請求項1記載の方法であって、メッセージ時間は意図された受信時間であり、かつ、メッセージはメッセージが送信された時間に第1の処理の局所仮想時間を指示する値も含んでいる方法。

【請求項12】 請求項11記載の方法であって、意図された受信時間および送信の仮想時間が同一の値を有する方法。

【請求項13】 1処理内の並行動作の一致性を保証するための方法であって、

処理変数の新バージョンが生成される際に、処理仮想時間および生成動作の識別性を、生成時間が当該変数のすべての現行バージョンの生成時間よりも大きくなければならないような変数に関連づけるステップと、

処理動作が変数を読み出す際に、その動作が実行される処理仮想時間に等しいかまたはそれより小さい仮想時間を有する変数の最新バージョンをその動作に読み出させ、かつ、動作の識別子および実行される仮想時間をその変数バージョンに関連するリストに加えるステップと、

変数のバージョンがいずれかの既存のバージョンよりも小さい仮想時間を持って生成された場合、当該の既存のバージョンを生成または読み出したすべての動作をロールバックさせるステップとを含み、さらに、

変数の特定のバージョンを生成した動作だけがそのバージョンを修正することができ、当該の修正はそのバージョンが生成された同一の処理仮想時間にのみ行うことができ、その変数バージョンに関するリーダリストで識別されるいずれの動作も当該変数バージョンが修正される際にロールバックされるものである方法。

【請求項14】 並行処理を実行する1コンピュータシステムにおける処理間通信を管理するための方法であって、

各処理の局所仮想時間を維持するステップと、

メッセージが2処理間で渡される際にそのメッセージ内に当該メッセージの要求受信時間を含ませるステップ

と、
受信メッセージが処理されるために処理によって選択される際にメッセージの状態がその処理の現在の状態と一致するかを判断するステップと、
メッセージおよび処理の状態が一致しない場合に処理を一致した状態までロールバックさせるステップとを含む方法。

【請求項１５】請求項１４記載の方法であって、処理の状態が、その処理の仮想時間および、処理変数バージョンを含むように定義される方法。

【請求項１６】請求項１５記載の方法であって、メッセージの状態は、そのメッセージの仮想受信時間が処理の現在の仮想時間よりも大きい、または、メッセージを処理するために必要な処理変数のバージョンがその処理内のそれらの変数の現在のバージョンに等しい場合に、処理の状態に一致しているものである方法。

【請求項１７】請求項１６記載の方法であって、処理の状態はさらに、処理の状態がメッセージの状態と一致していなければ満足化を検査される、処理ガード述語によって定義されるものである方法。

【請求項１８】請求項１７記載の方法であって、ガード述語は、処理の状態がメッセージの状態と一致している場合でも満足化を検査されるものである方法。

【請求項１９】請求項１４記載の方法であって、各メッセージはまたそのメッセージが送信される際に送信側処理の仮想時間を含み、かつ、各メッセージの仮想送信時間はそのメッセージの要求受信時間よりも小さいかまたは等しいものである方法。

【請求項２０】請求項１４記載の方法であって、処理に送信されたメッセージは、各メッセージで定義された要求受信時間の優先順に待ち行列に入れられ、その優先順に処理によって処理されるために選択されるものである方法。

【請求項２１】請求項２０記載の方法であって、すべての処理のすべての仮想時間、すべての処理待ち行列に残っているすべての未処理メッセージの仮想送信時間および、処理間を移行中のすべてのメッセージの仮想送信時間のうちの最小のものとして、大域的仮想時間が定義されるものである方法。

【請求項２２】請求項２１記載の方法であって、大域的仮想時間よりも小さい仮想時間を有するすべてのメッセージおよび中間的な退避された処理状態が、システムおよび再利用される各自の記憶空間によって放棄することができるものである方法。

【発明の詳細な説明】

【０００１】

【産業上の利用分野】本発明は、電子計算機システム、より詳しくは１コンピュータシステム内の並行処理の動作を制御する方法に関する。

【０００２】

【従来の技術およびその問題点】コンピュータのアプリケーションは、一連の論理処理がそのアプリケーションに要求する作業を分割する場合、並行同時的に実行する。処理は、多重プログラミング環境におけるプログラム、多重プロセッサまたは多重コンピュータのシステムのプロセッサの活動、他のトランザクションと並行同時的にデータベースにアクセスするトランザクション、または、目的向き計算における目的などである。

【０００３】並行処理は、密結合または疎結合であると分類することもできる。密結合処理は、共通にアクセスされた記憶空間の変数を共用することによって相互作用する。疎結合処理は、各処理が自己の記憶空間に単独にアクセスしながら、処理間でメッセージを渡すことによって相互作用する。

【０００４】疎結合処理の場合、メッセージは非同期的に渡すことができるので、処理間での情報の単方向転送が可能である。メッセージは同期させて渡すこともでき、この場合、単方向または双方向の情報転送が可能である。

【０００５】同期化の一般的方法は、ペシミスティックまたはオプティミスティックのいずれかに分類できる。ペシミスティックな方法は、並行演算の間違った順序づけが絶えず発生しないようにするものである。これは、システムの大域的状態がある演算を実行する前にその演算と一致しているようにすることによって達成できる。オプティミスティックな方法は、局所的に使用可能な情報にもとづいて順序づけを実行し、後に演算の順序づけが間違っていることが分かった場合に、以前に一致していた状態へのロールバックを行う。ペシミスティックな方法は、演算を実行する前に大域的なシステム状態を検査する時間を比例増加的に費やし、演算を繰り返す必要はない。オプティミスティックな方法は、演算を実行する前に局所的なシステム状態を検査する最小限の時間を費やすが、場合によってはロールバックが発生した後にいくつかの演算を繰り返し実行しなければならないこともある。

【０００６】並行処理の同期化を容易にするために多数の基本的な方法が使用されている。これらの方法には、ガード、データバージョン、および、事象インデックスを使用することが含まれる。ガードは処理の各動作に関する述語である。ある動作のガードは、その動作の実行が許可される前に真でなければならない。

【０００７】データバージョンの使用は、並行計算の順序が正しい計算順序に一致するように、並行計算によって利用されるデータに対して行われるタギング変更を含む。処理は、それらが作成しアクセスするデータのバージョンに関連づけられる。この方法の使用の例に、データベースのためのマルチバージョン並行性制御がある。

【０００８】タイムスタンプやシーケンス番号などの事象インデックスは、それぞれの順序が事象の因果的順序

5

に一致するように並行計算の事象にタグを付けるために使用される。これらのインデックスは、事象を処理と関係づけることによって並行計算を同期化するために使用できる。このような方法の一例が、D. R. Jeffersonによって開発され、“Virtual Time”（David R. Jefferson, ACM Transactionson Programming Languages and Systems, Volume 7, No. 3, July 1985, pp. 404－425）で説明された“タイムワープ”機構である。この機構は、オプティミスティックな方法による非同期式メッセージパッシングモデルにもとづいている。各局所的処理は関係づけられた局所仮想時間を持っている。処理間で渡されたメッセージは、それらが受信されるべき仮想時間を刻印される。処理がメッセージを受信すると、メッセージの受信時間がその処理の現在の仮想時間に満たない場合、処理はすでに、メッセージが届くはずであった時点を過ぎて進んでいる。このような事象では、処理はメッセージの受信時間よりも前の仮想時間にロールバックされる。これは、多数の動作の取消しを含み、その後、処理は再始動し、受信されたメッセージが作用する。

【0009】“タイムワープ”機構に関する問題の一つは、その効率が本来求められるべき効率よりも著しく低くなり得ることである。その処理の現在の仮想時間よりも小さい受信時間を持つメッセージが受信されると常に、処理はロールバックされる。ある場合には、処理がこのメッセージを処理した後に再始動すると、計算は以前同様に正確に進行する。言い換えれば、メッセージの内容および、その処理は、ロールバックしたいずれの動作に対しても影響しなかったことになる。処理のロールバックおよびその動作の再実行は費用がかかるので、実行される不要なロールバック量を最小限にするように並行処理を同期させるための方法を付与することが望ましいであろう。

【0010】上述と同様の考慮は、1処理内の並行動作がデータを共用する場合にも生じる。通常、1処理内の並行動作が不一致状態にならないように各種のロック方式が使用されている。1処理内の共用変数についてのタイムスタンプ順序づけの利用は、“タイムワープ”機構に関する上述と同様の非効率を生ずる。1処理の動作を実行するために必要な合計時間と実行されなければならないロールバック量の双方を最小限にするように処理の並行動作を同期させるための方法を付与することが望ましいであろう。さらに、そうした方法が、上述の課題に配慮した処理間同期方法と両立し矛盾しないことも望ましいであろう。

【0011】

【問題点を解決するための手段】従って、本発明の目的は、コンピュータシステムにおける並行処理間の同期および状態の一致性を保証するための方法を提供することである。本発明の第2の目的は、データの不一致の発生を最小限にする局所的に使用可能な情報にもとづいて処理が進行できるようにし、かつ、不一致が生じた場合には必要な再計算の量を最小限にする、そうした方法を提供することである。本発明の第3の目的は、データを共用する並行動作の1処理内の同期を行うために使用できるような方法を提供することである。

6

【0012】従って、本発明によれば、コンピュータシステムにおける並行処理を同期させるための方法が得られる。各処理は仮想時間を有する。メッセージが受信され、処理にって作用される際、これらのメッセージはその処理の現在の仮想時間に等しいかそれより大きい受信時間を有している場合、状態の一致性が仮定できる。ガード検査が使用可能であれば、ガードは、好ましくは、メッセージが作用される以前にすでに満たされていなければならない。メッセージの受信時間が現在の仮想時間に満たない場合、さらに状態の一致性の検査が行われる。このような一致性検査は変数バージョンおよびガードの一致性の検査を含むことができる。処理状態がメッセージの内容と一致すれば、そのメッセージは現在の仮想時間がメッセージの受信時間よりも大きい場合でも処理されることができる。同様の一般的方法は、共用変数が仮想タイムスタンプや変数バージョンによって状態一致性を検査される、並行動作を有する1処理内でも使用することができる。

【0013】

【実施例】図1について説明する。処理P，QおよびRの同時実行が図式的に示されている。矢印10は処理Pの進行を、同様に、矢印12は処理Qの、矢印14は処理Rの進行をそれぞれ示している。これらの各線10，12および14の下方への動きは、各処理によって遂行された作業量が増えたことを表現する。これらの時間線10，12および14上の丸点は、1処理内における事象、または動作を表し、波線16，18，20および22は処理間で送られたメッセージを表す。各メッセージは、送信側処理の局所仮想時間に送信され、受信側処理の局所仮想時間に受信される。例えば、メッセージ16は、事象P1で示された処理Pの局所仮想時間に送信され、事象Q4で示された処理Qの局所仮想時間に受信されている。

【0014】各処理間のメッセージの送信はこれらの処理内の各事象間の優先順位関係を確定する。処理時間線10，12および14ならびにメッセージ線16，18，20および22に沿って、ある事象から別の事象へ1経路を描くことができれば、これらの処理は規定の順序で実行されなければならない。例えば、事象P1は事象Q4よりも前に実行されなければならず、事象R1は事象P2よりも前に実行されなければならない。2事象間に経路がまったく描くことができなければ、それらの

事象はまったく同時並行であり、いずれかの順序で実行されることになる。例えば、事象Ｐ１は、事象Ｒ１，Ｑ１，Ｑ２，Ｑ３およびＲ４のいずれとも同時並行である。事象Ｑ４は事象Ｐ２およびＲ２と同時並行である。

【００１５】図２は、本発明に従って並行性制御を行うために使用されるメッセージ２４の内容を示す。本発明による処理同期化には要求されない、付加的な制御信号および情報は、含めることは可能であるが、示されてはいない。各メッセージは、送信側処理の識別子２６および受信側処理の識別子２８を有している。メッセージが送信された時間の送信側処理３０の局所仮想時間が含まれる。要求受信時間３２がメッセージに付加され、メッセージが含む何らかのデータ３５がそれに続く。

【００１６】メッセージ２４の受信時間３２は、送信側処理の送信時間３０に等しいかそれより大きい仮想時間として表される。受信側処理は受信時間３２にメッセージを読取り、処理しなければならない。送信側処理が、受信側処理の仮想時間がメッセージ２４の処理される時がどのような時であるかを考慮しない場合、ＮＵＬＬのような特殊な標識をメッセージ２４の受信時間領域３２に置くことができる。

【００１７】図３は、２処理間のメッセージの渡り方を示している。送信側処理３４はメッセージ３６を生成し、図２に関して述べたように情報を満たす。このメッセージ３６はその後、システムによって付与されたいずれかの機構によって受信側処理３８に転送される。多重プロセッサシステムまたは複数コンピュータリンケージの場合では、それぞれ、メッセージ３６は、システムバスまたはネットワークによって渡すことができる。並行処理がすべて単一のプロセッサで実行される場合、メッセージ３６は通常、大域的システムエリアに置かれた情報によって転送される。

【００１８】受信側処理３８は、通常、メッセージ３６が到着してすぐにメッセージを作用させることはない。代わりに、メッセージ３６は待ち行列４０に入れられる。待ち行列４０は、着信メッセージがメッセージ受信時間３２によって順序づけられる優先待ち行列である。受信側処理３８は、待ち行列４０からメッセージを選択する準備ができると、最小要求受信時間３２を持った未処理メッセージを選択する。

【００１９】受信側処理３８は、標識線４２で示された局所仮想時間を有している。処理３８が待ち行列４０からメッセージを受け取るごとに、処理はその局所仮想時間４２をメッセージの受信時間３２に向けて進める。処理３８の局所仮想時間４２がメッセージ３６の受信時間３２に満たない時点でメッセージ３６が待ち行列４０に着信した場合、メッセージは、矢印４４および４６で示される将来に処理されるようにする位置で待ち行列４０に入れられる。

【００２０】時には、メッセージ３６が矢印４８で示したような局所仮想時間４２にも満たない受信時間３２を有することがある。こうした場合、受信側処理３８の仮想時間は、メッセージ３６をすでに処理すべきであった時点を過ぎてしまっていることになる。前述の“ワープタイム”機構のような、従来技術によるシステムでは、受信側処理３８の局所仮想時間はメッセージ３６の受信時間３２以前の値までロールバックされなければならない。これは、受信時間３２に一致する局所仮想時間と現在の局所仮想時間４２との間に実行されたすべての動作を取り消すことを包含する。

【００２１】ロールバックを実行する処理は、受信側処理３８を以前の退避された状態まで戻し、その間に送信されていたあらゆるメッセージを取り消すことを伴う。処理の出力待ち行列に転送されたすべてのメッセージのコピーの退避方法および、ロールバックの事象でそうしたすでに送信されていたメッセージを取り消すための除去メッセージの転送方法を含む、この実行方法に関する詳細は、前述のＤ．Ｒ．Ｊｅｆｆｅｒｓｏｎによる論文“Ｖｉｒｔｕａｌ　Ｔｉｍｅ”に詳述されている。大域的仮想時間５０は、通常、待ち行列４０でいずれかの処理を待っている、または、現在システム内を通過中の、最も古い未処理メッセージと同程度以上に古い仮想時間である。Ｊｅｆｆｅｒｓｏｎの参考文献に述べられているように、大域的仮想時間５０は、ロールバックが生じる時点を過ぎた仮想時間を表し、すべての処理は、大域的仮想時間より以前のメッセージおよびその間に退避された処理状態を放棄することができる。

【００２２】各処理は互いに独立した仮想時間で動作しているので、メッセージ３６が受信側処理３８の現在の局所仮想時間４２より以前の受信時間３２を持つことはまれではない。これは、処理３８がメッセージ３６の受信時間３２より前の仮想時間にロールバックされなければならないことになり、これは相当に時間・資源集約的な手続きである。多くの場合、ロールバックは、メッセージ３６の内容がロールバックの間に行われなかった動作にまったく影響していなかったので、必要でない場合さえある。

【００２３】図４は、メッセージが作用されるべきかロールバックが必要かを判断するために待ち行列４０のメッセージを取り出す際に受信側処理３８により利用される、好ましい手順のフローチャートである。最初のステップ５２は、受信側処理３８が待ち行列から次のメッセージを選択することである。選択されたメッセージは、まだ作用されていないいずれかの未決定のメッセージのうち最も早い受信時間３２を有するような、待ち行列４０のメッセージである。その後、メッセージの受信時間Ｔ（Ｍ）がその処理の現在の局所仮想時間Ｔ（Ｐ）であるか否かを判断するために比較５４が行われる。メッセージの受信時間が現在の局所仮想時間よりも大きい場合、局所仮想時間は進められ、メッセージの受信時間に

等しく設定される５６。メッセージの時間が局所仮想時間に満たない場合、処理は自動的にロールバックされることはなく、その代わり、メッセージ内の参照されるいずれかの変数Ｖ（Ｍ）が受信側処理３８内の現在の処理変数バージョンＶ（Ｐ）と異なっているかどうかを判断するために検査が行われる５８。当業で公知のように、これらの処理変数バージョンは、それが必要になる場合、以前の状態へのロールバックを可能にするために処理によって保持される。ステップ５８では、受信メッセージを処理するために必要な変数が処理の現在の状態に一致しているかどうかを判断するために検査が行われる。

【００２４】ステップ５８の試験に対する答えが肯定であれば、処理を初期状態にロールバックする必要はない。メッセージの処理に必要な変数は、受信側処理３８が後の仮想時間に移行されている間に変更されておらず、ロールバックの間の一時的なすべての動作を取り消すことは不要であろう。処理の現在の状態は、そのメッセージに関する限り、以前の仮想時間のその状態と一致しており、それにより制御はステップ６０に渡されることができる。

【００２５】ステップ６０は、受信側処理３８に付与されたメッセージに対応するいずれかのガードの述語が満たされているかどうかを判断するための検査ステップである。ガードは、規定の要求条件が満たされるまで１以上の動作の処理を遅らせるために使用される、当業で公知の述語である。ガードがステップ６０に達した時に満足されていれば、制御はステップ６２に渡り、受信メッセージを処理するために必要な動作が実行される。ガードが満たされていなければ、例外が発生する６４。好ましくは、例外ハンドラはただちにエラーを指示することはない。ガードは、未着信メッセージにより生じたロールバックの結果として満たされるようにすることができる。そのようなロールバックは大域的仮想時間より遅いいずれかの仮想時間に対して生じることができるので、例外ハンドラは、好ましくは、大域的仮想時間が例外が発生した局所仮想時間に等しいまたはそれより大きくなるまで、エラーの信号を発しない。この時点では、いずれの処理も大域的仮想時間より以前の時間にロールバックさせることができないので、ガードを満たすことができなかったことによるエラーは取り消すことができず、エラーの信号が発信される。

【００２６】ステップ５８からのＮＯ分岐によって指示された、処理変数の状態が一致しない場合、処理動作ガードは、ステップ６０におけると同様に検査される６６。動作ガードが満足されない場合、ステップ５４および５８で検査された相対仮想時間および処理バージョンは無関係となり、動作が実行される６２。制御がステップ６６に達して、ガードが満たされていない場合、処理の局所仮想時間は、メッセージの受信時間Ｔ（Ｍ）より前の値にただちにロールバックされる６８。

【００２７】図４で述べた３種類の試験は、ある程度独立して取り扱うことができる。従って、ステップ５４の仮想時間の比較、ステップ５８での処理変数バージョンの比較、ステップ６０および６６で示されたガードの検査は必ずしもすべて実行される必要はない。全処理状態のこれらの３部分のそれぞれを検査することは、システムまたはアプリケーション設計者の要求によって、個別に作動させたり作動させないようにすることができる。

【００２８】変数バージョンおよびガードの試験が使用禁止となっていれば、ステップ５８，６０および６６は実行されない。こうした事象では、ロールバックは、メッセージの受信時間が処理の現在の仮想時間よりも小さい場合に自動的に実行される。これは、前述の“タイムワープ”機構に極めて類似の同期機構をもたらす。仮想時間の比較５４および処理変数の比較５８の試験が使用禁止となっていれば、制御はステップ６０に直接渡る。これは、当業で公知の方法に類似の方法で動作ガードを単純に検査するシステムをもたらす。システムが、仮想時間が使用禁止となっていて、こうした方法で作動している場合、ガードが初期に満たされていない場合に例外ハンドラによって検査される大域的仮想時間はまったくない。こうした事象では、例外ハンドラは、単に、ガードが絶えず満たされているかどうかを判断するために待っているにすぎない。ガードがずっと満たされるようになれば、対応する動作が実行される６２。ガードがまったく満たされないようになれば、システムが非活動状態になった場合にエラー状態の信号が発信できる。

【００２９】他の検査の組合せは、設計されるアプリケーションまたはシステムに対する希望によって、適切に行うことができる。３の独立した検査が行うことができるので、検査を行うべきかどうか判断するために使用できる、８の組合せが可能である。図４で示したように３の検査全部を使用可能にすることは、実行されなければならないロールバック数を最小限にするオプティミスティックな非決定性システムをもたらす。

【００３０】図５は、図４のフローチャートで説明した規則を用いた受信側処理によるメッセージ処理の単純な例である。リアルクロックタイムは図５で上から下へ増加する。列７０は、４の異なるメッセージの着信を示す。各メッセージは単一の文字によって識別されており、各メッセージの要求仮想受信時間が示されている。

【００３１】列７２は、受信メッセージに応答するために必要な動作を実行するために受信側処理によって費やされた時間を示す。この単純な例では、受信側処理は、逐次動作を実行できるにすぎない。しかし、一般には、基本システムがこうした処理を支援していれば、受信側処理が動作を並行して実行することは可能である。

【００３２】列７４は、各入力メッセージを処理するために必要な動作の実行において更新されなければならな

い変数を示している。列７６は、各メッセージが処理されている間の処理の仮想時間を示す。この局所仮想時間は単一メッセージの処理中には変更されない。列７８は受信待ち行列４０の現時点でのメッセージおよび、それらの優先順位を示す。

【００３３】時間８０において、メッセージＡが着信し、ただちに受信側処理によって処理される。局所仮想時間は、このメッセージの要求受信時間と同じ値まで進められる。メッセージＡが処理されている間に、メッセージＢおよびＣがそれぞれ、時間８２および８４に着信する。列７８に示すように、メッセージＣは、その２８という要求受信時間が３０というＢの要求受信時間よりも小さいために、メッセージＢより前に待ち行列に置かれる。メッセージＡの処理は時間８６に完了する。

【００３４】時間８８に、待ち行列の次のメッセージが処理のために取り出される。これはメッセージＣであり、その結果、メッセージＢだけが待ち行列に残される。メッセージＣの要求受信時間は２８なので、局所仮想時間は２８に設定される。時間９０に、メッセージＤが着信し、メッセージＢの前に待ち行列に入れられる。メッセージＣの処理が時間９２に完了し、時間９４に次のメッセージＤが処理のために取り出される。

【００３５】この時、メッセージＤの要求受信時間はその処理の現在の仮想時間よりも小さい。しかし、処理は必ずしも仮想時間２６にロールバックされるとは限らない。列７４に示すように、メッセージＡを処理している間に変数ＸおよびＹが更新され、メッセージＣを処理している間に変数Ｚが更新される。変数の更新は、新しい処理変数バージョンの生成を要求するが、それらを読取ることは生成を求めないので、従って新バージョンは更新された変数についてのみ生成される。列７４に示すように、変数ＺはメッセージＤの処理においては使用されないので、結果的に変数のバージョンは一致している。言い換えれば、メッセージＣの処理による変数Ｚの更新は、メッセージＤの処理に影響しない。これは、メッセージＤが仮想時間２６までロールバックすることなく処理できることを意味する。

【００３６】時間９６にメッセージＤの処理が完了する。その後、メッセージＢが待ち行列から取り出され、時間９８から処理される。メッセージＢの処理は時間１００に完了する。メッセージＤの処理がいずれかの方法で変数Ｚの値にアクセスした場合、Ｚの最新の値は、メッセージＤが処理されているはずであった仮想時間２６の時点で正しくないものであったであろう。この場合、２つの方法で取り扱うことができる。一つは、単に局所処理を仮想時間２６までロールバックさせ、メッセージＣの処理をやり直すことである。しかし、好ましくは、変数の値は必要な場合にロールバックを実行するために各バージョンについて退避されているので、メッセージＤの処理は単に変数Ｚの以前のバージョンを参照できるだけかもしれない。このロールバックの回避は、ＺがメッセージＤの処理中に読み出されるだけで書き込まれない場合にのみ使用できる。メッセージＤの処理が変数Ｚまたは、メッセージＣの処理によって読み出されたいずれかの変数を変更させる場合は、処理は仮想時間２６にロールバックされるべきである。例えば、メッセージＣの処理が、以前の仮想時間にメッセージＤの処理において変更されているはずであった、変数Ｘの値を読み出す場合、メッセージＣの処理はやり直されなければならない。

【００３７】変数のバージョンにもとづいてロールバックを行うべきか否かの上述の決定は、当然、図４のステップ６６に示したガードをうまく検査することによって無効にすることができる。前述の通り、メッセージ２４の受信時間３２は送信時間３０に等しいかまたはそれより大きくなければならない。受信時間３２が送信時間３０より大きければ、（仮想時間に関して）非同期のメッセージの渡しが処理間で生じる。送信時間３０と受信時間３２が等しければ、（仮想時間に関して）同期的なメッセージの渡しが生じる。これは、各処理の状態によって定義された仮想空間内のある時点で、メッセージの送信処理および受信処理が同一仮想時間に同一状態に達することを意味する。これは、Ａｄａのようなある種のプログラム言語におけるランデブーを実行する技法と類似である。

【００３８】このような同期通信によって、送信側処理は、受信側処理が同一の仮想時間に同期されるまで待つことができる。これは、受信側処理がその仮想時間をメッセージの要求受信時間まで進めるか、または、そうした仮想時間へのロールバックを許可するかのいずれかによって生じることができる。受信側処理はその後、要求される場合、送信側処理に返信メッセージを生成することができる。これが生じると、その２処理は、データ交換を実行するために結合または同期される。同期化時に実行されるいずれかの事象が発生した後、２処理は独立した実行を再開できる。

【００３９】図５に関して説明したように、受信側処理の仮想時間は、別様ではメッセージの仮想受信時間によって要求されるであろうロールバックを変数バージョンの等しいことが防止するのであれば、メッセージの要求受信時間より大きいことも可能である。しかし、そうした状況は、同期メッセージが渡された場合に２の処理の仮想時間を同期させることに一致しない。従って、好ましい実施例では、処理変数バージョンの同等またはガードの満足化にかかわらず、同期メッセージのメッセージ時間へのロールバックが必ず実行される必要がある。これにより、受信側処理は、受信側処理の仮想時間がロールバックされたので、同期メッセージと同一の仮想時間を持つ返信メッセージを送信することができる。

【００４０】ロールバックが発生しなければならない場

合を判断するために仮想時間および処理バージョンを使用する方法は、処理内の並行動作の活動を同期させるために単一処理内で使用することもできる。処理変数の新しいバージョンが生成される場合、それが生成された仮想時間およびそれを生成した動作は、そのバージョンによって記録される。同一または大きい仮想時間を有する処理内の動作だけがいずれかの所与のバージョンを読み出すことができる。あるバージョンを読み出した動作すべてのリストは、それが必要になった場合にロールバック手順を単純にするために、各バージョンによって保持される。

【００４１】図６について説明する。本発明に従って処理変数バージョンを記録するためのデータ構造が示されている。処理変数の現在のバージョン１０２は、その生成の仮想時間のポインタ１０４、それを生成した動作の識別子１０６を含んでいる。さらに、リーダリスト１０８も変数１０２に付加されており、変数１０２を読み出す各動作についての動作および仮想時間を識別する。同一変数のその前のバージョン１１０も、仮想時間１１２およびその生成の生成動作１１４ならびに、変数の以前のバージョン１１０を読み出した全動作のリーダリスト１１６を含んでいる。同様に、同一変数のさらに前のバージョン１１８も、生成時間１２０、生成動作識別子１２２およびリーダリスト１２４を含んでいる。

【００４２】１処理内の並行動作の正しい同期化を保証するには、変数にアクセスするための以下の規則で十分である。

【００４３】（１）時間１０４と同じかそれよりう大きい仮想時間を有する動作だけがその変数の現在のバージョン１０２を読み出すことができる。以前の仮想時間を有する動作はその変数の以前のバージョンを読み出せるだけである。各動作は、必ず、読み出されることが可能な変数の最新のバージョンを読み出す。

（２）処理変数の新バージョンが生成される場合、生成時間は、同一変数のすべての既存のバージョンの生成時間よりも大きくなければならない。大きい仮想時間を有する現行バージョンを生成したすべての動作はロールバックされなければならない。さらに、これらのバージョンのリーダリスト上のすべての動作もロールバックされなければならない。新バージョンが生成される場合、代わってこの新バージョンを読み出すべき直前のバージョンのリーダリスト上のすべての動作もロールバックされなければならない。こうした場合の例は、１５という仮想時間に別の動作によって読み出されていた１０という仮想生成時間を持つ現行の変数バージョンであろう。このような変数の新バージョンが仮想時間１３に生成される場合、仮想時間１５に以前のバージョンを読み出した動作は、仮想時間１３に生成された、より新しいバージョンを読み出すことを可能にするためにロールバックされなければならない。

（３）各処理変数バージョンは、ブロック１０６，１１４および１２２で識別されたように、それを生成した動作によってのみ修正することができる。こうした修正は、生成時間と同一の仮想時間にのみ行うことができる。変数のバージョンがそのように修正された場合は必ず、その変数のリーダリストの全動作は、それらが変数を読み出すための等しいかそれ以後の時間を持っており、新しく修正された変数を読み出すことが可能とならなければならないので、ロールバックされる。

【００４４】図６に関して説明したロールバックは、図４に関して概説した試験によって生じるロールバックに付加されるものである。上述のような生成時間および変数バージョンの保持は、共用変数が使用される場合に１処理内の並行動作の同期化を保証する。この並行処理バージョンの保持はまた、図５の例で説明した処理の現在の仮想時間より以前の要求受信時間を有するメッセージを処理する際に、変数の読出し要求を満たすために、処理が以前のバージョンを使用できるようにさせる。仮想時間が前述のように変数バージョンと組み合わされ、処理間通信だけでなく、並行動作間で変数を共用する処理内通信に適用された場合、システム内で完全な状態の一致性が維持される。

【００４５】上述の技法は並行目的向き計算にも適用できる。こうした計算では、各目的は論理的処理に、目的の属性は処理変数に、そして、目的の方法は処理動作に対応する。そのようなシステムでは、仮想時間、目的属性バージョンおよび目的方法予備条件が並行目的向き計算を同期させるために使用される。これは、上述の仮想時間、処理変数バージョンおよび、従来の並行処理のためのガードの使用と直接的に類似であり、同様に解釈できる。

【００４６】並行処理および１処理内の並行動作を同期させるための上述のシステムおよび方法は、並行処理を包含する通常のシステムの効率を向上させる。発生しなければならない処理のロールバック数は、処理の現在の仮想時間より小さい要求受信時間を有するメッセージの受信ごとに処理をロールバックさせる必要が必ずしもないため、低減される。さらに、ガードが満たされていないために動作が遅延される場合、後の要求されるロールバックは、その処理の状態についてそれ以上の変更が行われず、メッセージもまったく生成されないので、著しく容易になる。メッセージは並行処理の間を同期させても非同期でも渡すことができる。同期メッセージは２の処理が仮想時間でその活動を同期するようにさせる。アプリケーションプログラマは、それらの動作がプログラマにとっては知らなくてもよい形でオペレーティングシステムによって取り扱われることができるので、仮想処理時間および変数バージョン検査の機能に配慮する必要がない。上述の技法によって正しい同期化が保証されるので、アプリケーションプログラマは、他に先行して実

15

行される必要があるすべての動作がその通りに実行されることを想定できる。

【００４７】本発明を好ましい実施例によって詳細に説明したが、本発明の精神および範囲を逸脱することなく形式および細部の各種変更が可能であることは当業者によって理解されるであろう。

【図面の簡単な説明】

【図１】並行処理間で渡されるメッセージを図示するタイミング図。

【図２】並行処理間で渡されるメッセージに含まれる情報を示すデータ構造の説明図。

【図３】１処理への１メッセージの着信を示す高水準ブロック図。

【図４】処理間メッセージの受信時に処理により実行される状態一致性検査の好ましい集合を示すフローチャート。

【図５】図４の方法の１例を示すタイミングチャート。

【図６】１処理内の並行動作への好ましい方法の適用において作成されるデータ構造の集合。

16

【符号の説明】

５２　受信側処理が待ち行列から次のメッセージを選択する

５４　メッセージの受信時間Ｔ（Ｍ）とその処理の現在の局所仮想時間Ｔ（Ｐ）とを比較する

５６　局所仮想時間をメッセージの受信時間に等しく設定する

５８　メッセージ内の参照変数Ｖ（Ｍ）と受信側処理内の現在の処理変数バージョンＶ（Ｐ）とを比較検査する

６０　ガード述語が満たされているかどうかを判断するために検査する

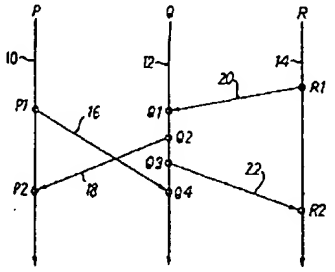
６２　受信メッセージを処理するのに必要な動作を実行する

６４　例外が発生する

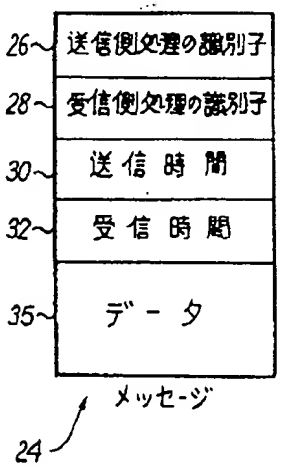
６６　ガード述語が満たされているかどうかを判断するために検査する

６８　処理の局所仮想時間をメッセージの受信時間Ｔ（Ｍ）より前の値にロールバックする

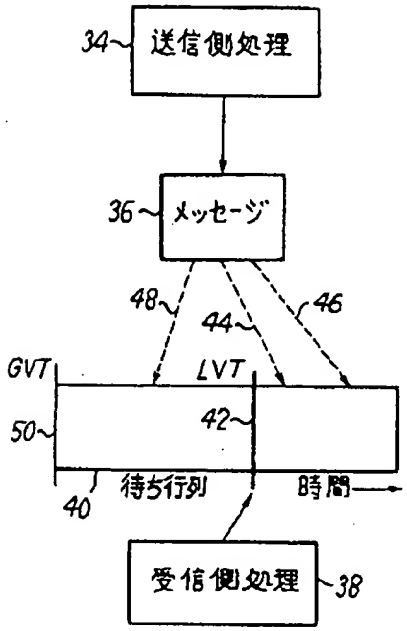
【図１】



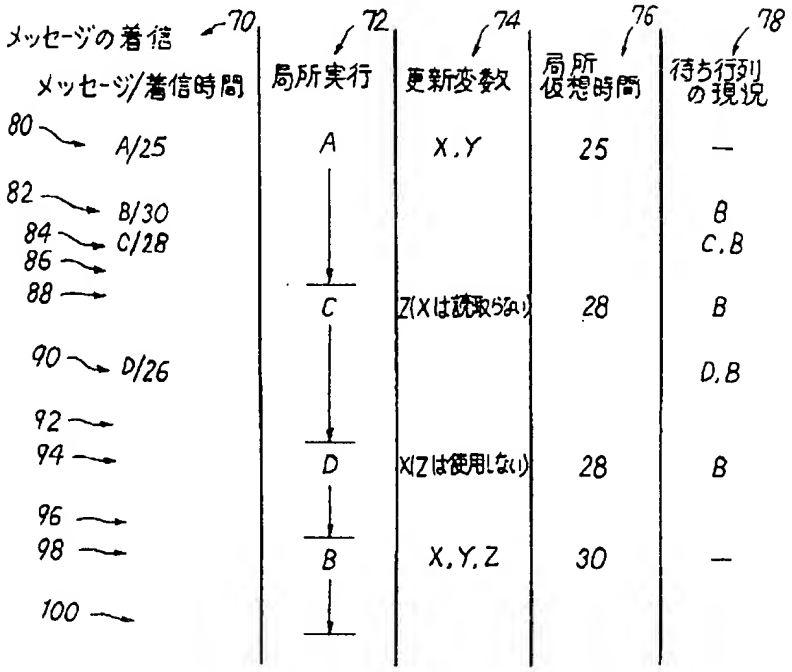
【図２】



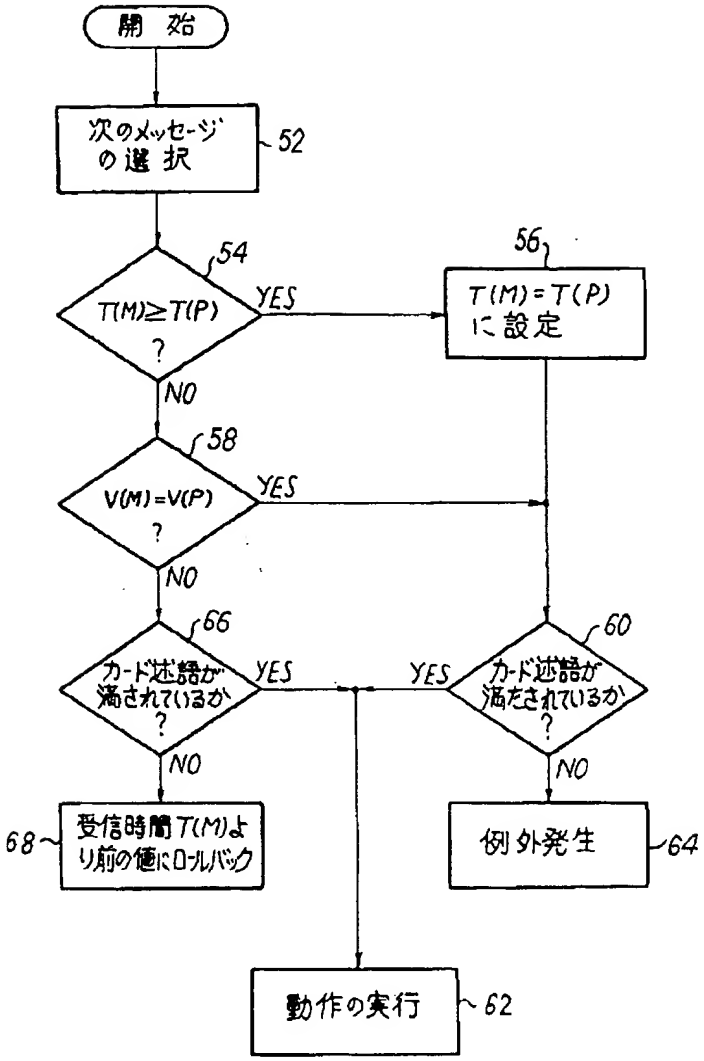
【図3】


34
送信側処理
36
メッセージ
48
44
46
GVT
LVT
42
50
待ち行列
時間
40
受信側処理
38

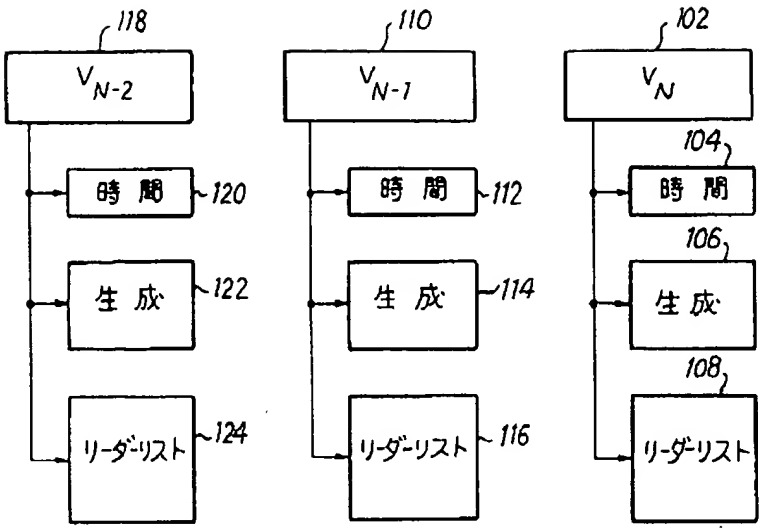
【図5】


メッセージの着信
メッセージ/着信時間
70
72
局所実行
74
更新変数
76
局所仮想時間
78
待ち行列の現況
80
A/25
A
X,Y
25
—
82
B/30
B
84
C/28
C,B
86
88
C
Z(Xは読取らない)
28
B
90
D/26
D,B
92
94
D
X(Zは使用しない)
28
B
96
98
B
X,Y,Z
30
—
100

【図4】


開始
次のメッセージの選択
52
54
T(M)≧T(P)?
YES
56
T(M)=T(P)に設定
NO
58
V(M)=V(P)?
YES
NO
66
カード述語が満されているか?
YES
YES
60
カード述語が満たされているか?
NO
NO
68
受信時間T(M)より前の値にロールバック
例外発生
64
動作の実行
62

【図6】



フロントページの続き

(72)発明者　ダニエル、ティー、チャン
アメリカ合衆国カリフォルニア州、サン、
ノゼ、オリーブ、ブランチ、レーン、1126